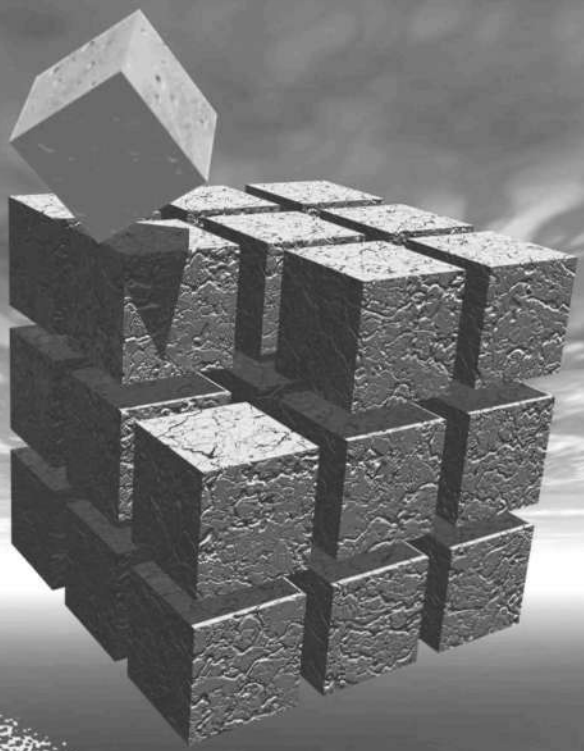
COLLADA + MPEG-4 OR X3D + MPEG-4

Marius Preda,
Ivica Arsov, and
Francisco Morán

An Overview of 3-D Graphics Assets and Applications Standards



© DIGITAL STOCK

We analyzed the three main open standards dealing with three-dimensional (3-D) graphics content and applications, X3D, COLLADA, and MPEG-4, to clarify the role of each with respect to the following criteria: ability to describe only the graphics assets in a synthetic 3-D scene or also its behavior as an application, compression capacities, and appropriateness for authoring, transmission, and publishing. COLLADA could become the interchange format for authoring tools; MPEG-4 on top of it (as specified in MPEG-4 Part 25), the publishing format for graphics assets; and X3D, the standard for interactive applications, enriched by MPEG-4 compression in the case of online ones.

Since 1963, when Sutherland created the first computer program for drawing [1], the field of synthetic graphics has followed, as many others involving computers, more or less the same exponential growth foreseen by Gordon E. Moore in 1965 with respect to the advances in semiconductor industry. In the early years, the progress was pushed up by scientific interest, the real boom being reached when the need for special effects and 3-D graphics content came from the film industry and later from the video games. Of those two industries, the former leads to the development of technologies for the production of high-quality images, and the latter, which has already outgrown the former, leads to the efficient techniques for real-time processing and rendering.

Even from the beginning, the strategy for developing 3-D graphics solutions was based on three requirements: faster, cheaper, and better (image quality). Traditionally, the 3-D graphics chain was focused on the two ends, production and consumption, each one having its own strong requirements. Additionally, in many applications such as games, dealing with complex 3-D environments, there is an iterative loop involving content creators and programmers or beta testers, making sure that the content fits a specific end-user terminal configuration. The complexity of creating 3-D graphics content leads to the coexistence of several authoring tools, such as the ones shown in Figure 1, each specialized in one or more particular tasks.

TO BUILD A MOBILE APPLICATION, A DEVELOPER HAS TO CONSIDER DIFFERENT HARDWARE CONFIGURATIONS AND PERFORMANCES, DIFFERENT OPERATING SYSTEMS, DIFFERENT SCREEN SIZES, AND INPUT CONTROLS.

Therefore, it is possible to (have to) use several tools to create one single 3-D asset, and interoperability is an issue, so either a unique representation standard is used by all tools or data format converters must be provided in each of them and for all others in the ideal case.

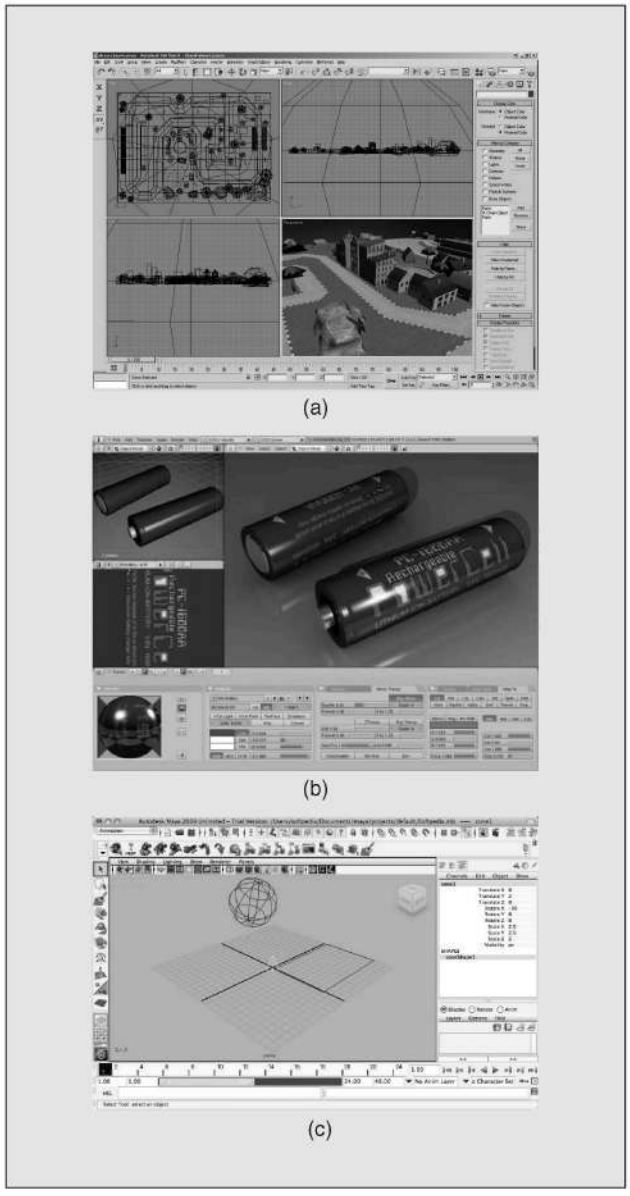


FIGURE 1 Examples of authoring tools to create synthetic 3-D content: (a) 3ds Max, (b) Blender, and (c) Maya. (Courtesy of Autodesk and Blender Foundation.)

The rest of this article is organized as follows. The next section provides an overview of 3-D graphics standards giving a historical perspective and stressing the content distribution requirements that a good standard should meet. The "Standards for Individual 3-D Graphics Assets and Scene Graph" section focuses on how different standards support the representation and coding of isolated 3-D assets as well as their integration in a scene, thanks to the widely accepted concept of scene graph. The following section deals with normalization at the application level, showing how it is possible to combine different standards to create rich scenarios. Each of these sections addresses the particular case of the mobile platforms in its last subsection.

3-D Graphics Standards Overview

Historical Perspective

In the last decade, several efforts have been made to develop a unique data format for interchanging assets between authoring tools. In the open standards category, X3D [2] (based on VRML97) and COLLADA [3] are the best known, the latter probably being the most adopted by current tools, although still far from being the Holy Grail of the 3-D graphics production industry. While COLLADA concentrates on representing 3-D objects or scenes, X3D pushes the standardization further by addressing as well user interaction, thanks to an event model in which scripts, possibly external to the file containing the 3-D scene, may be used to control the behavior of its objects.

One of the characteristics of both X3D and COLLADA is the data representation based on XML. Indeed, using XML allows a very easy extension of the structure, as one can impregnate data with meaning and purpose, implicitly upgrading it to be self-contained information. XML is an appropriate mechanism to express dynamic 3-D scenes because even noninteractive ones usually contain complex structures of heterogeneous data organized in different kinds of graphs (for both the scene as a whole and its possibly many objects) and containing elements of inherently different nature (geometry, appearance, animation, lights, viewpoints, etc.).

On the consumption side, data should be as flat (i.e., close to the hardware) as possible to enable easy and rapid processing by the graphics cards. A lot of effort has been spent for optimizing the rendering step, one of the bottlenecks in the 3-D synthetic content pipeline. Better, faster algorithms are continuously proposed, and some of them have been directly implemented in the silicon, allowing rendering performances that some years ago nobody could have even dreamed of (see Figure 2). To normalize the access to different graphics cards, the low-level 3-D graphics application programming interfaces (APIs) such as OpenGL and Microsoft Direct3D were created and they keep evolving. Such APIs hide the different capabilities of

the many and wildly diverse graphics cards in the market (with and without specific hardware acceleration for 3-D visualization, with more or less texture memory, etc.) and offer the programmer a single set of rendering functions. Given a particular graphics card, some of these functions will translate to direct hardware calls and some will be executed in software, because all implementations of the API being used must support the full feature set. The higher level rendering engines such as OGRE [4] have been designed to enable working with objects and scene concepts, thus making it easier and more intuitive for developers to produce applications meant to be run on hardware-accelerated 3-D graphics cards. Such libraries provide an extra abstraction layer on top of the underlying system APIs such as Direct3D and OpenGL.

Content Distribution Requirements

Recent developments of collaborative sharing platforms (the well-known Web2.0) are efficient vehicles for pushing media over the Internet and introduce the additional need of compactness in data representation. Even in the case of games, which have traditionally been distributed in DVDs containing all assets, openness to new distribution channels is a must, either for entire game downloading before playing or for the streaming of assets during the match in the case of online games. Neither the formats used in the content creation phase nor the ones used for rendering is appropriate for transmission, because they usually satisfy none of the three requirements that are most important in this context:

- *compression*: to reduce the size of the transmitted data
- *streamability*: to enable players to start using the content before having downloaded it entirely
- *scalability*: to ease real-time adaptation of the content to a specific platform and network.

Generic compression on top of XML (achieved through entropy coding of the corresponding text file, thanks to gzip, WinRAR, or similar tools) usually reduces the data size only by a factor 10 [5], because it is just able to exploit the data structure redundancy, since the information represented by the data is not understood and hence cannot be exploited. Even worse, streaming and scalable coding are out of the question with generic compression methods because both require that the data semantics be understood to build hierarchical or at least progressive representations of the information, which can then be adequately packetized. Instead, specific 3-D graphics (lossy) compression techniques may exploit better spatial or temporal correlation present in the information and reduce the data size by a factor of more than 40 [6], [7] while possibly yielding progressive, scalable bitstreams suitable for streaming scenarios involving heterogeneous networks and/or terminals.

***MPEG-4 AFX** CONTAINS SEVERAL TECHNOLOGIES FOR THE EFFICIENT STREAMING OF COMPRESSED MULTITEXTURED POLYGONAL **3-D** MESHES THAT CAN BE EASILY AND FLEXIBLY ANIMATED.*

Built on top of VRML97, MPEG-4 contained, already in its first specifications from ten years ago [8], tools for the compression and streaming of 3-D graphics assets, enabling to describe compactly the geometry and appearance

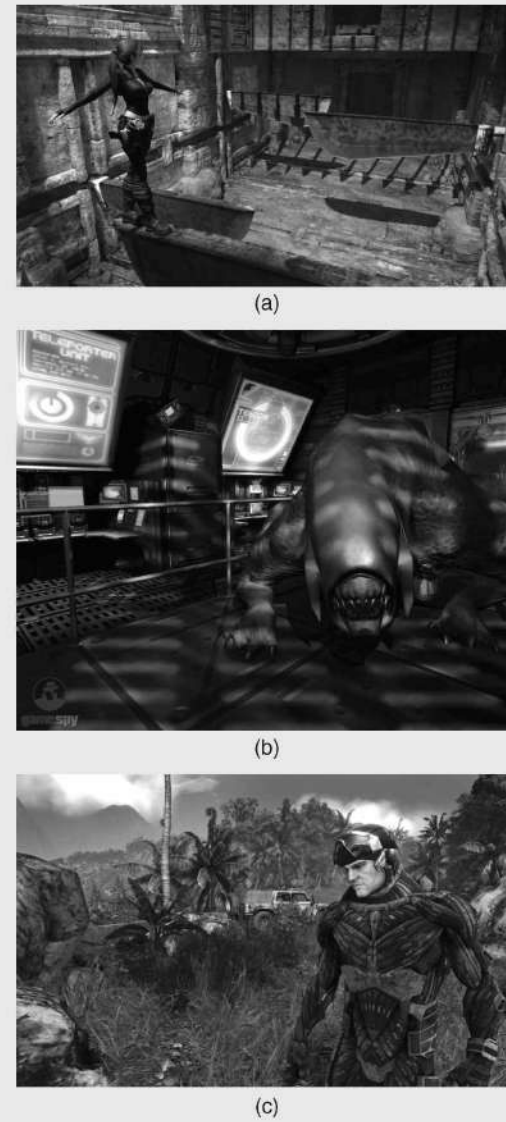


FIGURE 2 Examples of games using synthetic 3-D content: (a) Tomb Raider Underworld (developed by Crystal Dynamics and published by Eidos), (b) Crysis (Crytek and Electronic Arts respectively), and (c) Doom 3 (ID Software and Activision respectively). (Courtesy of Eidos, Electronic Arts and Activision.)

A TRIANGULAR 3-D MESH IS NOT ONLY THE EASIEST WAY TO DESCRIBE A STATIC SHAPE BUT ALSO ONE OF THE MOST COMMONLY USED.

of generic, but static objects, and also the animation of humanlike characters. Since then, MPEG has kept working on improving its 3-D graphics compression toolset and published three editions of MPEG-4 Part 16, Animation Framework Extension (AFX) [9], which addresses the aforementioned requirements within a unified and generic framework and provides many more tools to compress generic, textured, and animated 3-D objects more efficiently.

Mobile Platforms Market

The invasion of 3-D graphics, currently ongoing in the digital multimedia world, could not avoid the mobile devices. However, this trend is below expectations: some studies from 2001 had forecast a mobile games market of US$18 billion for 2007, whereas the real size was of only US$2.5 billion. Several reasons of different nature were proposed to explain the underperformance with respect to business maturity and usability, and one of the technological barriers was identified to be the fragmentation of the mobile phones market because of the massive device proliferation. To build a mobile application, a developer has to consider different hardware configurations and performances, different operating systems, different screen sizes, and input controls. This diversity adds extra time and costs to the development cycle of a 3-D application. To minimize them, different attempts to stabilize the technology are currently in progress. At the hardware level, Nokia proposes with the N-series and particularly with the N-Gage, a relatively powerful terminal, with the intention to become a dedicated mobile console for games and graphics applications. The same approach of considering the device as the stabilization point is followed by Apple with its iPhone. At a higher level, but still in the hardware realm, stabilization is ongoing for graphics chipsets. Major actors from the traditional desktop graphics chipsets such as ATI and NVIDIA propose low-consumption chips ready to be integrated in mobile devices. On the other hand, several layers of software such as graphics libraries (OpenGL ES and Direct3D mobile), virtual machines (Java with JSR-184, a.k.a. M3G), and dedicated rendering engines (MascotCapsule or X-Forge) are available.

Standards for Individual 3-D Graphics Assets and Scene Graph

Isolated Objects

The modeling of a single-3-D object begins with the description of its shape, which is then typically textured, to have it look more attractive or photorealistic, and possibly also animated, to have it look alive. There are several possible ways to model the shape of a static 3-D object of which the simplest is to describe only its bounding surface. In turn, there are several possible ways to approximate a 3-D surface, the simplest being to tile it with polygons—in fact, with triangles, the simplest polygons. A triangular 3-D mesh is not only the simplest and most compact way to describe a static shape [6], [7] but also still one of the most commonly used in practice, because it is the best choice regarding low-level standards, i.e., the ones closest to hardware. Indeed, triangular meshes are the typical output of 3-D-scanning systems and the input typically preferred by the most widely used 3-D graphics libraries such as OpenGL or Direct3D, because graphics cards are still designed to accelerate their rendering and not that of higher order approximations to surfaces.

With respect to de jure international standards, textured, animated polygonal meshes are supported by the two families of standards published until now by International Organization for Standardization (ISO) for describing and coding 3-D scenes: VRML97 (and its successor X3D) and MPEG-4. Both consider the classic superficial approximation to a 3-D shape based on the indexed face set (IFS) paradigm and offer the possibilities of mapping one or more textures (likely compressed according to a picture coding standard: JPEG[2000], PNG, GIF, etc.) onto the IFS, and of animating/deforming its geometry over time. In particular, MPEG-4 AFX [9] contains several technologies for the efficient streaming of compressed multitextured polygonal 3-D meshes that can be easily and flexibly animated, thanks to the bone-based animation (BBA) toolset.

Looking now at the de facto formats, triangular meshes are also increasingly common, although animation is clearly eased by higher order surface approximations based for instance on polynomial/rational patches. All commercial 3-D modeling applications (3ds Max, Maya, etc.) are able to handle triangular meshes but unfortunately store them in their own proprietary formats, all different from one another, and so direct interoperability is not possible. When it comes to dynamic meshes, the problem is even worse: although all 3-D software packages support the higher order approximations mentioned earlier and most follow the skeleton, muscles, and skin animation paradigm and have some kind of inverse kinematics (IK) engine, the differences between their proprietary formats to store the data related to animation are much wider, rather logically because the problem is much more complex than that of merely describing a static IFS.

This is because, traditionally, no company having played an important role in the 3-D graphics market (e.g., Nintendo, Sega, Sony, or Microsoft in the case of

videogames; Autodesk or Avid for modeling/animation software; and NVIDIA or ATI for hardware) has shown almost any interest in sticking to any standard format for interchanging 3-D graphics assets. However, it seems that this tendency to isolationism and security through obscurity could change soon or, rather, is changing already, thanks to the creation of the Khronos group (www.khronos.org) in 2000. Khronos is an industrial consortium of more than 100 companies, which fosters the adoption of open and royalty-free standards for the description, coding, and rendering of 3-D content. This consortium has taken care of the maintenance of OpenGL and produced already new versions of it and other related specifications, but its flagship is probably COLLADA, a 3-D content interchange scheme (as opposed to a mere file format) that is very likely to become the de facto standard in this market.

Objects Within a Scene

Once modeled separately, the individual 3-D objects that will compose a global scene must be combined together, the concepts of object hierarchy and scene graph being very helpful and widely used to position objects relative to each other and to describe their interrelations.

Since 1999, when the first version of MPEG-4 was published, the MPEG committee has had a continuous activity on compressing 3-D graphics assets and scenes. MPEG-4 offers a rich set of tools that may be classified with respect to data type. The most generic format, called binary format for scene (BIFS), allows compressing a scene graph expressed in a VRML97-like fashion. The downside of being a generic tool is that BIFS cannot completely exploit the redundancy of specific data such as meshes or animation curves. To overcome this, MPEG-4 defines methods for each kind of graphics primitive: for shapes, 3-D mesh coding (3DMC) and wavelet subdivision surface (WSS); for textures, visual texture coding (VTC) as well as the native support of PNG and JPEG[2000]; and for animations, coordinate (CI), orientation (OI), and position interpolators (PI), BBA, and frame-based animated mesh compression (FAMC).

The penetration of COLLADA in the market, as well as the adoption of XML for X3D, made the MPEG committee issue the new Part 25 of the MPEG-4 set of specifications, with the goal of offering an architectural model able to accommodate a third-party XML description of the scene graph with (potential) binarization tools and with MPEG-4 3-D graphics compression tools. The advantages of such an approach are the use of powerful specific compression tools and the generality of graphics primitives representation. Hence, compression tools specified in other parts of MPEG-4 would not only be applied only to the scene graph defined by MPEG-4 Part 1, Systems, but also to any scene graph definition taken from an external standard. The bitstreams obtained using the proposed architectural model are MP4 formatted and advanced

COLLADA COULD BECOME THE INTERCHANGE FORMAT FOR AUTHORING TOOLS; MPEG-4 ON TOP OF IT, THE PUBLISHING FORMAT FOR GRAPHICS ASSETS.

compression features such as support for streaming and progressive decompression, provided by many MPEG-4 specific tools, are preserved with the new model, because it decouples the decoding of the scene graph from that of the object graph. Typically, when consuming content compliant with MPEG-4 Part 25, a player first builds the scene graph based on an XML description, then connects the elementary streams to the adequate decoders, and, finally, renders the results taking into account the synchronization information carried by the time stamps. There is no need to rebuild the original XML file before rendering.

The current model can accommodate any scene graph and graphics primitive representation formalism. The only requirement for this representation is that it should be expressed in XML. Any XML schema may be used: currently, the ones of COLLADA and X3D are supported, meaning that MPEG-4 Part 25 specifies the correspondence between each XML element and the compression tool handling it.

The well-known model of Web2.0 platforms for video and images (YouTube and Flikr), consisting of a database of media assets and presentation engines, was applied to 3-D graphics assets. One of the most known databases, Google 3-D Warehouse, uses zipped COLLADA to represent the assets. An experimental database, designed for research purposes and available online at www.MyMultimediaWorld.com [10], uses MPEG-4 as a common representation format for all media types, including 3-D graphics assets. The player is embedded in the Web browser navigator as illustrated in Figure 3.

Mobile Platform Implementation

Implementations of COLLADA [11] and MPEG-4 [12] players exist as well for the S60 platform of Nokia phones. Recent MPEG-4 implementations as the one shown in Figure 4 indicate reasonable decoding times for static objects of around 10,000 vertices and animated objects of more than 3,000 vertices, in all cases the rendering frame rate being above 20 frames/s on a Nokia N95 cell phone.

The advantages of using this standardization approach at the level of graphics assets were demonstrated recently in the context of a research project on online gaming funded by the European Union, for which a scheme was devised for adapting 3-D graphics based on scalable coding, real-time simplification, and remote rendering [13]. Thanks to it, 3-D objects were transmitted and adapted between different platforms during game play (Figure 5).

FIGURE 3 Presentation engine of MyMultimediaWorld.com based on an MPEG-4-compliant player. (Courtesy of Institut TELECOM.)

FIGURE 5 Screen shots from both the PC and cell-phone versions of a multiplatform game with inherently scalable 3-D content. [Courtesy of the European research project FP6-IST-1-507926, OLGA (a unified framework for online gaming.)]

Standards for 3-D Graphics Applications

One of the biggest steps forward of MPEG-4 with respect to its predecessors, MPEG-1 and MPEG-2, was the definition of BIFS as a part of the system information, which lies on top of all media data and acts as an intermediate layer between media data and the final displayed content. It gives a flexible way to manipulate various types of audio–visual media in an MPEG-4 interactive scene, to schedule events, coordinate multimedia objects both in the temporal and spatial domains, process interactivity, etc. BIFS is largely based on VRML97. It is a very efficient, binary-encoded version of an extended set of VRML97, and it is similar to VRML97 (and thus X3D), in that it includes tools to handle the user input (sensors), propagate actions in the scene graph (routes), and specify application behavior (scripts and Java applets). It is therefore possible to program complete two-dimensional (2-D) and/or 3-D games and interactive applications [14] that are interpreted by a compliant player or browser, as illustrated in Figure 6.

Strategies for Mobile Platforms

The complexity of using scripts or applets embedded in the content itself makes it suitable for powerful platforms but less appropriate for mobile ones. Other than the traditional

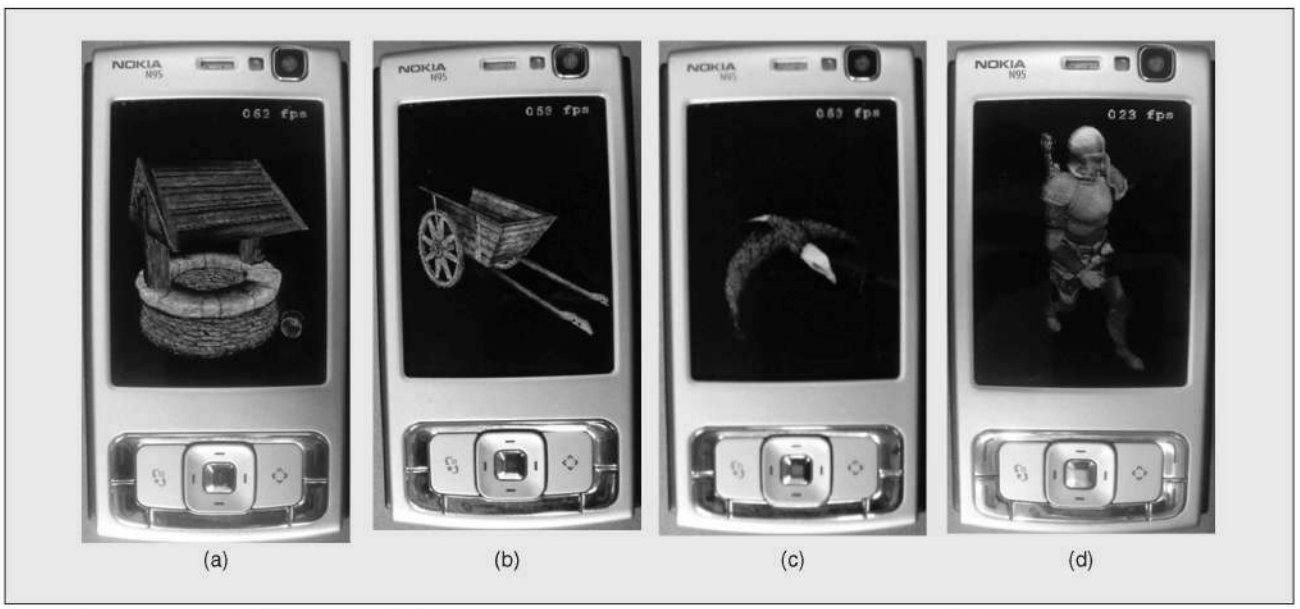


FIGURE 4 Snapshots for (a), (b) static and (c), (d) dynamic 3-D objects displayed on a Nokia N95 phone. (d) The hero model, one of the most complex objects we have tested: it consists of nine disconnected subobjects, 3,587 vertices, 6,846 triangles, nine textures, 63 bones, and 31 animation frames. The decoding times and rendering frame rates are as follows: (a) 1.6 s, 63 frames/s; (b) 1.7 s, 59 frames/s; (c) 3 s, 62 frames/s; (d) 86 s, 23 frames/s. (Courtesy of Institut TELECOM.)

approach of interpreting locally the 3-D graphics primitives and computing the pixel maps, the intrinsically online nature of mobile phone allows the alternative approaches of remote computing and/or rendering. The latter consists of interpreting the graphics primitives on a server (much more powerful than the mobile phone) and then send the result as a video stream to the mobile device [15]. Through the back-channel, the user interaction with the terminal (keyboard and screen touch) is directly transmitted to the server. The only processing requirement on the client side is to have a terminal able to decode and display the video stream. The major limitation of this solution is the need of significant bandwidth, making it feasible only when the network connection can ensure good transmission quality. On the server side, this approach requires significant computational power and leads to poor performances if too many users are loading the system simultaneously. Additionally, the graphics card(s) of the server must also be shared between the users. As an alternative, the rendering can be distributed on dedicated rendering clusters, thus supporting more clients.

Another possibility is to execute the game logic on the server and the rendering on the mobile terminal as illustrated in Figure 7. In addition, to ensure that there are no situations with high peaks of data transmission during the game play, the entire initial scene and media layers can be encapsulated and transmitted before starting the game. This operation is performed by using the MPEG-4 file format, able to represent complex 3-D graphics scenes with associated multimedia data. In addition, it is possible to adapt the graphics content for a specific terminal [13], allowing for the best possible tradeoff between performance and quality. By building the architecture on top of the MPEG-4 standard, the communication between the rendering engine and the game logic follows a standardized formalism. This allows for creating them separately, so that developers can generate game logic for different games that will use the same rendering engine without taking into account how the latter is implemented. On the other hand, the developers of the rendering engine can create optimized solutions for a specific platform without knowing the game logic.

The communication is based on a higher level control mechanism: the graphic primitives can be grouped and controlled as a whole by using few parameters. The MPEG-4 standard allows any part of the scene to be loaded, changed, reorganized, or deleted. As an example, the same game can be played on a rendering engine that supports 3-D primitives,

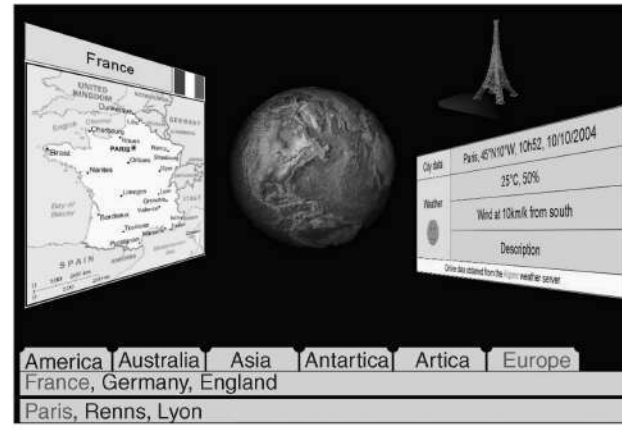


FIGURE 6 A geographic encyclopedia visualized by an MPEG-4 player. (Courtesy of Institut TELECOM.)

probably accelerated by dedicated hardware, and simultaneously on a rendering engine that only supports 2-D primitives. This flexible approach allows distribution of the games on multiple platforms without the need to change the code of the game logic. Another advantage is the possibility to improve the game logic without additional cost to the client, allowing easy bug-fixing and addition of features and optimizations. Because the game logic can be hosted on a server that has much better performances than the client terminal, it can implement more advanced features that normally will not be available on the client terminal. These features may include artificial intelligence, advanced physics, and any other techniques usually requiring a large computational power. These improvements will not change anything on the user side, so the user will be able to play more complex games on the same terminal. The rendering engine can also be integrated in the

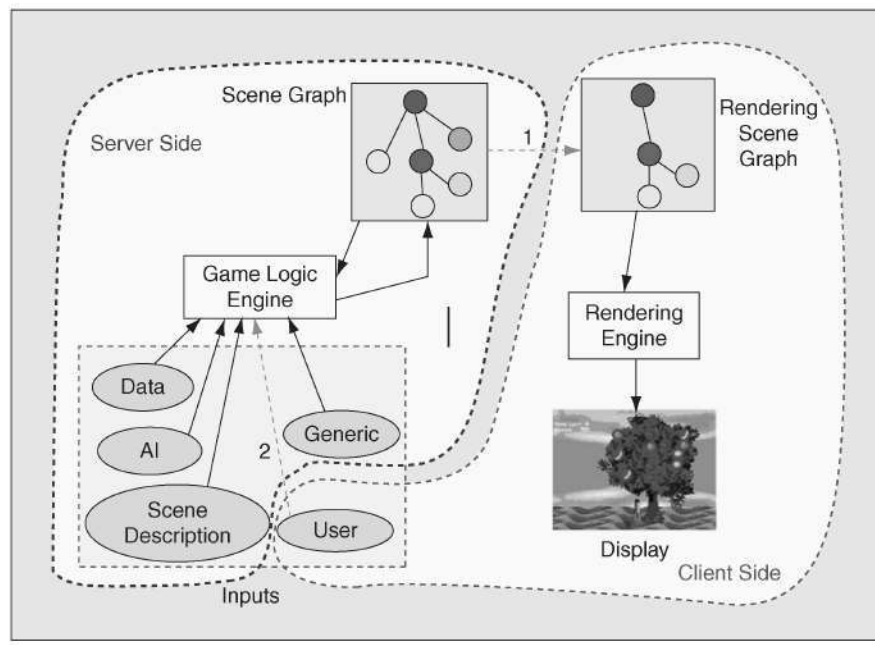


FIGURE 7 An alternative client–server architecture for executing complex 3-D applications (including games) on mobile platforms. (Courtesy of Institut TELECOM.)

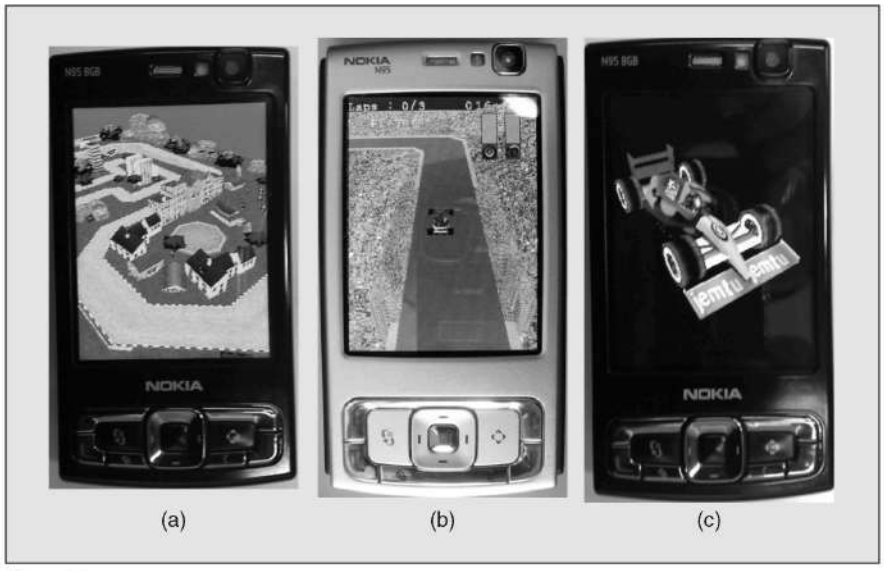


FIGURE 8 (a)–(c) Snapshot of the race game visualized with the MPEG-4 player and using a remote server for computing the game logic. [Courtesy of the Institut TELECOM research project JEMTU (JEux sur Mobiles: Technologies et Usages, i.e., Games on Mobile: Technologies and Usage.)]

firmware of the device, allowing the manufacturer to optimize it for the specific hardware. Even more, the different decoders supported by MPEG-4 (video, audio, 3-D graphics, and animation) can be integrated in a dedicated hardware to improve performance.

The flexibility of the architecture illustrated in Figure 7 makes it appropriate for powerful terminals as well. In these conditions, it is possible to download the game logic server software, install it on the terminal, and use it to play the game locally. However, to maintain this approach effective, the games should be designed from the start having in mind the requirements and the restrictions. A snapshot of a game built with this architecture in mind is presented in Figure 8.

TABLE 1 ***Comparative evaluation of the classical method (local game play) versus the proposed one (client–server approach).***

	Local Game Play	***Client–Server Approach***
Advantages		
SW development	Game must be compiled for each specific terminal.	Game logic is compiled once for the dedicated server, and rendering is ensured by a standard MPEG-4 player, optimized for each specific terminal.
Advanced game features	Reduced due to terminal limitations.	Modern game features such as artificial intelligence (AI) can be supported by choosing high-end servers.
Rendering frame rate	Lower: terminal processing power is shared between game logic and rendering.	Higher: terminal only performs occasional, asynchronous scene, graph decoding, and rendering.
Game state consistency in multiplayer games	A complex scheme must be designed and implemented to exchange synchronous signals between terminals.	Synchronization is inherently ensured by the server that controls at each step the scene graph of each terminal.
Security	Games can be easily copied.	Game only starts after connection to the server, where an authorization process may be easily established and updated.
Maintenance and game updates management	Patches/updates must be compiled for each version of the game.	Patches/updates need to be installed only in the server.
Disadvantages[1]		
Network		Permanent network connection is a must.
Network latency		Game experience decreases if the loop user interaction—server processing—update commands—rendering is not fast enough.[2]
Adaptation of existing local games to proposed architecture		Access to the game source code is a must.

[1]With respect to local game play.
[2]The notion of fast is very game-specific: for first-person shooters, the mentioned loop must be performed in 70 ms or less; for race and strategy games, 150 and 500 ms are enough, respectively.

Table 1 summarizes the pros and cons of the two strategies discussed earlier: the more classical one in which the game logic is computed locally on the (mobile) terminal, and the client–server approach we propose, in which it is computed remotely by the server.

Conclusion

Irony has it that the good thing about standards is that there are so many to choose from. While there is a rather unanimous feeling that standards are in general beneficial for business and end users, in the case of 3-D graphics, there was until recently a strong tendency for reinventing the wheel every time a new authoring tool was born: it is much easier to format the data in a proprietary way, that one may change afterward, than to spend time in understanding a standard and sticking to it forever. That is why there are currently dozens of de jure and de facto standards for representing exactly the same thing: textured and possibly animated 3-D shapes (usually approximated by triangular meshes) positioned relative to each other in a synthetic scene. Even taking into account only ISO standards, this trend was identified as wrong in the past (VRML97 and MPEG-4 are around ten years old), but the recent publication of COLLADA may have changed things. In this new picture, each of these three specifications has their own role: COLLADA could become the interchange format for authoring tools; MPEG-4 on top of it, as specified in MPEG-4 Part 25, the publishing format of graphics assets; and X3D (the newest version of VRML97), the format for interactive applications, enriched by MPEG-4 compression in the case of online ones.

Graphics on mobile platforms are currently repeating the history of personal computers in the early 1990s with some differences: hardware development speed is faster; an increasingly sophisticated infrastructure provides both for high-speed networks and common servers to assist the mobile devices; and there is a clearly improved know-how on content representation formalisms. All these facts combined make it possible to forecast a brilliant future for mobile devices in the field of 3-D graphics content and applications.

Author Information

Marius Preda (marius.preda@it-sudparis.eu) received his engineering degree in electronics from University Politehnica, Bucharest, Romania, in 1998 and a Ph.D. degree in mathematics and informatics from University Paris V—René Descartes, Paris, France, in 2001. He is currently an associate professor at Institut Telecom. His research interests include generic virtual character definition and animation, rendering, low bit-rate compression and transmission of animation, and multimedia composition and standardization. He is the chair of the 3-D graphics subgroup of ISO's Moving Picture Experts Group (MPEG) and is involved in several projects at the institutional, national, and European levels as a technological leader and an MPEG-4 expert.

Ivica Arsov received his engineering degree in electrical engineering at the Faculty of Electrical Engineering, Skopje, Macedonia, in 2005. He is currently a Ph.D. student at the Institut Telecom/Telecom and Management SudParis within the ARTEMIS Department. His research interests include 3-D graphics on mobile phones, both storage and representation, streaming of 3-D content, and architectures for 3-D games on mobile devices. He also participates in MPEG-4 standardization activities.

Francisco Morán received his engineering and Ph.D. degrees in telecommunication from Universidad Politécnica de Madrid (UPM), Spain, in 1992 and 2001, respectively, and joined its Image Processing Group in 1992. Five years later, he started teaching at UPM, where he is a full-time associate professor since 2002. His research interests include 3-D object modeling, coding, transmission, and rendering and has been working for more than a dozen years in projects funded by the European Commission. He also belongs to MPEG since 1996, where he has edited several 3-D graphics-related MPEG-4 specifications and is the head of the Spanish Delegation.

References

[1] I. E. Sutherland, "Sketchpad: A man-machine graphical communication system," Ph.D. thesis, MIT, Cambridge, MA, Jan. 1963.

[2] Web3D Consortium and ISO/IEC JTC1/SC24, "ISO/IEC 19775-1:2004," a.k.a., "X3D (Extensible 3D), Part 1, Architecture and base components, ed. 1," ISO, Nov. 2004 (ed. 2 published in July 2008).

[3] R. Arnaud and M. C. Barnes, *COLLADA: Sailing the Gulf of 3D Digital Content Creation*, Natick, MA: A.K. Peters, 2006.

[4] G. Junker, *Pro OGRE 3D Programming*. New York: Apress, 2006.

[5] C. J. Augerl, D. A. Bulutoglu, B. E. Mullins, R. O. Baldwin, and L. C. Baird, III, "An analysis of XML compression efficiency," in *Proc. ACM Workshop on Experimental Computer Science (ExpCS'07)*, June 2007, 12 pp.

[6] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3D mesh compression: A survey," *J. Vis. Commun. Image Represent.*, vol. 16, pp. 688–733, Apr. 2005.

[7] M. Avilés and F. Morán, "Static 3D triangle mesh compression overview," in *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Oct. 2008, pp. 2684–2687.

[8] MPEG (Moving Picture Experts Group, formally ISO/IEC JTC1/SC29/WG11), "ISO/IEC 14496-2:1999," a.k.a., "MPEG-4 Part 2, Visual, ed. 1," ISO, Jan. 1999 (ed. 3 published in May 2004).

[9] MPEG, "ISO/IEC 14496-16:2004," a.k.a., "MPEG-4 Part 16, AFX (Animation Framework eXtension), ed. 1," ISO, Feb. 2004, (ed. 3 published in Dec. 2009).

[10] B. Le Bonhomme, M. Preda, and F. Prêteux, "From MPEG-4 scene representation to MPEG-7 description," in *Multimedia Semantics—The Role of Metadata* (Studies in Computational Intelligence, vol. 101), M. Granitzer, M. Lux, and M. Spaniol, Eds. Berlin, Germany: Springer-Verlag, May 2008, pp. 25–44.

[11] K. Pulli, "New APIs for mobile graphics," in *Proc. SPIE Electronic Imaging: Multimedia on Mobile Devices II*, vol. 6074, SPIE, Jan. 2006, pp. 1–13.

[12] I. Arsov, M. Preda, and F. Prêteux, "A server-assisted approach for mobile-phone games," in *Mobile Multimedia Processing: Fundamentals, Methods, and Applications*, to be published.

[13] M. Preda, P. Villegas, F. Morán, G. Lafruit, and R. P. Berretty, "A model for adapting 3D graphics based on scalable coding, real-time simplification and remote rendering," *Vis. Comput.*, vol. 24, no. 10, pp. 881–888, Oct. 2008.

[14] S. M. Tran, M. Preda, F. Prêteux, and K. Fazekas, "Exploring MPEG-4 BIFS features for creating multimedia games," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, July 2003, vol. 2, pp. 429–432.

[15] I. Nave, H. David, A. Shani, A. Laikari, P. Eisert, and P. Fechteler, "Games@Large graphics streaming architecture," in *Proc. IEEE Int. Symp. Consumer Electronics (ISCE)*, Apr. 2008, pp. 1–4.